# COMPUTER SCIENCE

A.K. Watson Hall, 203.432.1246
http://cpsc.yale.edu
M.S., M.Phil., Ph.D.

**Chair**
Holly Rushmeier

**Directors of Graduate Studies**
Lin Zhong (lin.zhong@yale.edu)
Vladimir Rokhlin

**Professors** Dana Angluin (*Emerita*), James Aspnes, Dirk Bergemann,* Abhishek Bhattacharjee, Ronald Coifman,* Aaron Dollar,* Julie Dorsey, Joan Feigenbaum, Michael Fischer, Robert Frank,* David Gelernter, Mark Gerstein,* John Lafferty,* Rajit Manohar,* Vladimir Rokhlin,† Holly Rushmeier, Brian Scassellati, Martin Schultz (*Emeritus*), Zhong Shao, Avi Silberschatz, Daniel Spielman, Phillipp Strack,* Leandros Tassiulas,* Nisheeth Vishnoi, Y. Richard Yang, Lin Zhong, Steven Zucker†

**Associate Professors** Yang Cai, Amin Karbasi,* Theodore Kim, Smita Krishnaswamy,† Sahand Negahban,* Charalampos Papamanthou, Ruzica Piskac, Robert Soule, Jakub Szefer*

**Assistant Professors** Ian Abraham,* Kim Blenman,* Arman Cohan, Yongshan Ding, Benjamin Fisch, Tesca Fitzgerald, Julian Jara-Ettinger,* Anurag Khandelwal, Quanquan Liu, Tom McCoy,* Daniel Rakita, Katerina Sotiraki, David van Dijk,* Marynel Vázquez, Andre Wibisono, Alex Wong, Zhitao Ying, Manolis Zampetakis

**Senior Lecturers** James Glenn, Stephen Slade

**Lecturers** Timos Antonopoulos, Timothy Barron, Ozan Erat, Kyle Jensen,* Janet Kayfetz, Jay Lim, Dylan McKay, Cody Murphey, Sohee Park, Scott Petersen, Brad Rosen, Alan Weide, Cecillia Xie

* A secondary appointment with primary affiliation in another department or school.

† A joint appointment with another department.

## FIELDS OF STUDY

Algorithms and computational complexity, artificial intelligence, data networking, databases, graphics, machine learning, programming languages, robotics, scientific computing, security and privacy, and systems.

## RESEARCH FACILITIES

The department operates a high-bandwidth, local-area computer network-based mainly on distributed workstations and servers with internet connections. Laboratory contains specialized equipment for graphics, robotics, systems, and vision research. Various printers, including color printers, as well as image scanners, are also available. The primary educational facility consists of a large cluster of personal computers. This facility is used for courses and unsponsored research by computer science majors and first-year graduate students. Access to computing, through both the workstations and remote login facilities, is available to everyone in the department.

## SPECIAL REQUIREMENTS FOR THE PH.D. DEGREE

There is no foreign language requirement. To be admitted to candidacy, a student must

1. pass ten courses (including CPSC 690 and CPSC 691) with at least two grades of Honors, the remainder at least High Pass, including three advanced courses in an area of specialization;
2. take six advanced courses in areas of general computer science;
3. successfully complete a research project in CPSC 690, CPSC 691, and submit a written report on it to the faculty;
4. pass a qualifying examination in an area of specialization;
5. be accepted as a thesis student by a regular department faculty member;
6. serve as a teaching assistant for two terms; and
7. submit a written dissertation prospectus, with a tentative title for the dissertation.

Grades of Pass will not count toward the Ph.D. To satisfy the distribution requirement (requirement 2 above), the student must take one course in programming languages or systems, one programming-intensive course, two theory courses, and two in application areas. In order to gain teaching experience, all graduate students are required to serve as teaching assistants for two terms during their first three years of study. All requirements for admission to candidacy must be completed prior to the end of the third year. In addition to all other requirements, students must successfully complete CPSC 991, Ethical Conduct of Research, prior to the end of their first year of study. This requirement must be met prior to registering for the second year of study.

## MASTER'S DEGREES

**M.Phil.** See Degree Requirements under Policies and Regulations.

**M.S. (en route to the Ph.D.)** To qualify for the M.S., the student must pass eight courses at the 500 level or above from an approved list. An average grade of at least High Pass is required, with at least one grade of Honors.

**Terminal Master's Degree Program** Students may also be admitted to a terminal master's degree program directly. There are two options for the terminal master's degree:

- *Terminal Master's Degree Program (coursework-only option)* The requirements are the same as for the M.S. en route to the Ph.D. This program is normally completed in one year, but a part-time program may be spread over as many as four years.
- *Terminal Master's Degree Program (thesis option)* To qualify for the M.S. thesis option the student must (1) pass six courses at the 500 level or above from an approved list with an average grade of at least High Pass and with at least one grade of Honors; (2) complete a research thesis, generally in the second year; and (3) serve as a teaching assistant for four terms. This program is normally completed in two years.

Please use the links provided for additional information about the department, faculty, courses, and facilities online; You may also reach out to the departmental registrar at cs-admissions@cs.yale.edu.

## COURSES

**CPSC 510b, Physics Simulation for Movies** Theodore Kim
This course covers computational methods for simulating physics in movies. In particular, we learn state-of-the-art methods for simulating fluids (fire and water) and solids (muscles, clothing, and skin). The algorithms discussed span offline techniques suitable for movies and touch on how they can be adapted into real-time techniques for games. We cover finite difference and finite element representations and solver practicalities such as conjugate gradients, preconditioning, and Newton iteration. Prerequisite: *The student must have taken CPSC 578 or its equivalent*. Experience with linear algebra (e.g. MATH 222 or 225 equivalents), Newtonian physics (e.g. PHYS 171 or 181 equivalents), and two semesters of programming experience will be assumed. The student should be able to read and write in an imperative programming language like C/C++ or Java.

**CPSC 513a, Computer System Security** Timothy Barron
Overview of the principles and practice behind analyzing, designing, and implementing secure computer systems. The course covers problems that have continued to plague computer systems for years as well as recent events and research in this rapidly evolving field. Students learn to think from the perspective of an adversary, to understand systems well enough to see how their flaws could be exploited, and to consequently defend against such exploitation. The course offers opportunities for hands-on exploration of attacks and defenses in the contexts of web applications, networks, and system-level software. It also addresses ethical considerations and responsibilities associated with security research and practice.

**CPSC 516a, Lattices and Post-Quantum Cryptography** Katerina Sotiraki
This course explores the role of lattices in modern cryptography. In the last decades, novel computational problems, whose hardness is related to lattices, have been instrumental in cryptography by offering: (a) a basis for "post-quantum" cryptography, (b) cryptographic constructions based on worst-case hard problems, and (c) numerous celebrated cryptographic protocols unattainable from other cryptographic assumptions. This course covers the foundations of lattice-based cryptography from fundamental definitions to advanced cryptographic constructions. More precisely, we introduce the Learning with Error (LWE) and the Short Integer Solutions (SIS) problems and study their unique properties, such as the fact that their average-case hardness is based on the worst-case hardness of lattice problems. Next, we cover lattice constructions of advanced cryptographic primitives, such as fully homomorphic encryption and signature schemes. Finally, we introduce some notions of quantum cryptography and explore the role of lattices in this area. Overall, this course offers insights on the foundations and recent advancements in lattice-based cryptography. Prerequisites: CPSC 467/567 or equivalent and linear algebra.

**CPSC 517a, Advanced Topics in Cryptography: Cryptography and Computation** Charalampos Papamanthou
Traditional cryptography is mostly concerned with studying the foundations of securing communication via, for example, encryption and message authentication

codes. This class studies the applications of cryptography in securing *computation*. Topics include, but not limited to, fundamental results and most recent progress in oblivious computation and private information retrieval (PIR), zero-knowledge proofs, secure computation, consensus algorithms, searchable encryption, and lattice-based cryptography. The class focuses both on theory and applications. This is an advanced course, which requires mathematical maturity as well as comfort with programming. The course also assumes prior knowledge of fundamental notions in cryptography. Prerequisite: CPSC 467 or equivalent.

**CPSC 518b, Network System Design and Implementation: Internet Abstractions and Layer-7 Control** Y. Richard Yang
Network systems have become pervasive due to the success of the Internet. However, the design and implementation of network networked systems are challenging due to the fundamental black-box nature of the Internet. In this course, we study the design and implementation of network systems, with a focus on designing and implementing fundamental abstractions and application-layer control of the Internet. Taken concurrently with or after CPSC 323.

**CPSC 519b, Full Stack Web Programming** Alan Weide
This course introduces students to a variety of advanced software engineering and programming techniques in the context of full-stack web programming. The focus of the course includes both client- and server-side programming (and database programming), client/server communication, user interface programming, and parallel programming.

**CPSC 520b / ENAS 820b, Computer Architecture** Staff
This course offers a treatment of computer architectures for high-performance and power/energy-efficient computer systems. Topics include the foundations of general-purpose computing, including instruction set architectures, pipelines, superscalar and out-of-order execution, speculation, support for precise exceptions, and simultaneous multi-threading. We also cover domain-specific hardware (e.g., graphics processing units), and ongoing industry efforts to elevate them to the status of first-class computing units. In tandem, we cover topics relevant to both general-purpose and domain-specific computing, including memory hierarchies, address translation and virtual memory, on-chip networks, machine learning techniques for resource management, and coherence techniques. If time permits, we study the basics of emerging non-classical computing paradigms like neuromorphic computing. Overall, this course offers insights on how the computing industry is combating the waning of traditional technology scaling via acceleration and heterogeneity. Prerequisites: Courses similar to CPSC 323, 223, and 202. This is a programming-intensive course, so comfort with large programming projects is essential.

**CPSC 521a, Compilers and Interpreters** Jay Lim
Compiler organization and implementation: lexical analysis, formal syntax specification, parsing techniques, execution environment, storage management, code generation and optimization, procedure linkage, and address binding. The effect of language-design decisions on compiler construction.

**CPSC 522b, Operating Systems** Anurag Khandelwal
The design and implementation of operating systems. Topics include synchronization, deadlocks, process management, storage management, file systems, security, protection, and networking.

**CPSC 524b, Parallel Programming Techniques** Quanquan Liu
Practical introduction to parallel programming, emphasizing techniques and algorithms suitable for scientific and engineering computations. Aspects of processor and machine architecture. Techniques such as multithreading, message passing, and data parallel computing using graphics processing units. Performance measurement, tuning, and debugging of parallel programs. Parallel file systems and I/O.

**CPSC 526a, Building Distributed Systems** Y. Richard Yang
Ubiquitous services such as Google, Facebook, and Amazon run on the back of massive distributed systems. This course covers the fundamental principles, abstractions, and mechanisms that inform the design of such systems, as well as the practical details of real-world implementations. Technical topics covered include properties such as consistency, availability, durability, isolation, and failure atomicity; as well as protocols such as RPC, consensus, consistent hashing, and distributed transactions. The final project involves implementing a real-world distributed service.

**CPSC 527a, C++ Programming for Stability, Security, and Speed** Michael Fischer
Computer programming involves both abstraction and practice. Lower-level programming courses focus on learning how to correctly implement algorithms for carrying out a task. This course treats a computer program as an artifact with additional attributes of practical importance including execution efficiency, clarity and readability, redundancy, safety in the face of unexpected or malicious environments, and longevity—the ability to evolve over time as bugs are discovered and requirements change. This course is taught using modern C++.

**CPSC 529a, Principles of Computer System Design** Lin Zhong
Humans are stupid; computers are limited. Yet a collaboration of humans and computers has led to ever more powerful and complex computer systems. This course examines the limitations of humans and computers in this endeavor and how they shape the design, implementation, and evaluation of computer systems. It surveys the empirical knowledge reported by scholars and practitioners who overcome such limitations. The lectures, reading assignments, and classroom discussions travel through psychology and philosophy and revisit important results from theoretical computer science, with a goal of elucidating the rationales behind the best practices in computer systems research and development. Prerequisite: CPSC 323 or equivalent. Students should have the ability to write significant system programs in at least one system programming language (e.g., C, C++ and Rust).

**CPSC 530b, Formal Semantics** Zhong Shao
Introduction to formal approaches to programming language design and implementation. Topics include lambda calculus, type theory, denotational semantics, type-directed compilation, higher-order modules, and application of formal methods to systems software and Internet programming.

**CPSC 531a, Computer Music: Algorithmic and Heuristic Composition** Scott Petersen
Study of the theoretical and practical fundamentals of computer-generated music. Music and sound representations, acoustics and sound synthesis, scales and tuning

systems, algorithmic and heuristic composition, and programming languages for computer music. Theoretical concepts are supplemented with pragmatic issues expressed in a high-level programming language.

**CPSC 532b, Computer Music: Sound Representation and Synthesis** Scott Petersen
Study of the theoretical and practical fundamentals of computer-generated music, with a focus on low-level sound representation, acoustics and sound synthesis, scales and tuning systems, and programming languages for computer music generation. Theoretical concepts are supplemented with pragmatic issues expressed in a high-level programming language. Prerequisite: ability to read music.

**CPSC 537a, Database Systems** Avi Silberschatz
An introduction to database systems. Data modeling. The relational model and the SQL query language. Relational database design, integrity constraints, functional dependencies, and natural forms. Object-oriented databases. Implementation of databases: file structures, indexing, query processing, transactions, concurrency control, recovery systems, and security.

**CPSC 539a, Software Engineering** Timos Antonopoulos
Introduction to building a large software system in a team. Learning how to collect requirements and write a specification. Project planning and system design. Increasing software reliability: debugging, automatic test generation. Introduction to type systems, static analysis, and model checking.

**CPSC 540a, Database Design and Implementation** Robert Soule
This course covers advanced topics in Database Systems, explaining on the material covered in CPSC 437/537. Topics covered include complex data types, application development, big data, data analytics, parallel and distributed storage, parallel and distributed query processing, advanced indexing techniques, advanced relational database design, and object-based databases.

**CPSC 544b, Secure Decentralized Systems** Fan Zhang
Decentralized systems are computer systems composed of multiple autonomous agents. A notable example is the Internet, whose crucial services such as DNS and BGP are operated by autonomous organizations, realizing benefits such as strong resiliency. Excitingly, recent developments in Internet-scale consensus (in particular "blockchains"), cryptography, and hardware-assisted trustworthy computing have enabled new kinds of decentralized systems. This course studies the theory and practice of these recent developments. On the theory side, students read and discuss recent papers published in related areas. On the practice side, students examine representative state-of-the-art decentralized systems in the wild. This course is aimed at Ph.D. and M.S. students. It is highly recommended for students to have passed a computer security course like CPSC 513 and/or a cryptography course like CPSC 567.

**CPSC 546a, Data and Information Visualization** Holly Rushmeier
Visualization is a powerful tool for understanding data and concepts. This course provides an introduction to the concepts needed to build new visualization systems, rather than to use existing visualization software. Major topics are abstracting visualization tasks, using visual channels, spatial arrangements of data, navigation in visualization systems, using multiple views, and filtering and aggregating data. Case studies to be considered include a wide range of visualization types and applications in humanities, engineering, science, and social science. Prerequisite: CPSC 223.

**CPSC 547a, Introduction to Quantum Computing** Yongshan Ding
This course introduces the fundamental concepts in the theory and practice of quantum computing. Topics covered include information processing, quantum programming, quantum compilation, quantum algorithms, and error correction. The objective of the course is to engage students in applying fresh thinking to what computers can do. We establish an understanding of how quantum computers store and process data, and we discover how they differ from conventional digital computers. We anticipate this course will be of interest to students working in computer science, electrical engineering, physics, or mathematics. Students must be comfortable with programming, discrete probability, and linear algebra. Prior experience in quantum computing is useful but not required.

**CPSC 551b, The User Interface** David Gelernter
The user interface (UI) in the context of modern design, where tech has been a strong and consistent influence from the Bauhaus and U.S. industrial design of the 1920s and 1930s through the IBM-Eames design project of the 1950s to 1970s. The UI in the context of the windows-menus-mouse desktop, as developed by Alan Kay and Xerox in the 1970s and refined by Apple in the early 1980s. Students develop a detailed design and simple implementation for a UI.

**CPSC 552b / AMTH 552b / CB&B 663b / GENE 663b, Deep Learning Theory and Applications** Smita Krishnaswamy
Deep neural networks have gained immense popularity within the past decade due to their success in many important machine-learning tasks such as image recognition, speech recognition, and natural language processing. This course provides a principled and hands-on approach to deep learning with neural networks. Students master the principles and practices underlying neural networks, including modern methods of deep learning, and apply deep learning methods to real-world problems including image recognition, natural language processing, and biomedical applications. Course work includes homework, a final exam, and a final project—either group or individual, depending on enrollment—with both a written and oral (i.e., presentation) component. The course assumes basic prior knowledge in linear algebra and probability. Prerequisites: CPSC 202 and knowledge of Python programming.

**CPSC 554a, Software Analysis and Verification** Ruzica Piskac
Introduction to concepts, tools, and techniques used in the formal verification of software. State-of-the-art tools used for program verification; detailed insights into algorithms and paradigms on which those tools are based, including model checking, abstract interpretation, decision procedures, and SMT solvers.

**CPSC 555a, Economics and Computation** Yang Cai
A mathematically rigorous investigation of the interplay of economic theory and computer science, with an emphasis on the relationship of incentive-compatibility and algorithmic efficiency. Particular attention to the formulation and solution of mechanism-design problems that are relevant to data networking and Internet-based commerce.

**CPSC 562b, Spectral Graph Theory** Dan Spielman
An introduction to spectral graph theory motivated by computer science, covering advanced topics in linear algebra and exploring the combinatorial meaning of the eigenvalues and eigenvectors of matrices associated with graphs. Applications to

optimization, numerical linear algebra, error-correcting codes, pseodorandomness, and the discovery of graph structure. Prerequisites: linear algebra, graph theory, and comfort with proof-based math courses.

**CPSC 564a, Algorithms and their Societal Implications** Nisheeth Vishnoi
Today's society comprises humans living in an interconnected world that is intertwined with a variety of sensing, communicating, and computing devices. Human-generated data is being recorded at unprecedented rates and scales, and powerful AI and ML algorithms, which are capable of learning from such data, are increasingly controlling various aspects of modern society: from social interactions. These data-driven decision-making algorithms have a tremendous potential to change our lives for the better, but, via the ability to mimic and nudge human behavior, they also have the potential to be discriminatory, reinforce societal prejudices, violate privacy, polarize opinions, and influence democratic processes. Thus, designing effective tools to govern modern society which reinforce its cherished values such as equity, justice, democracy, health, privacy, etc. has become paramount and requires a foundational understanding of how humans, data, and algorithms interact. This course is for students who would like to understand and address some of the key challenges and emerging topics at the aforementioned interplay between computation and society. On the one hand, we study human decision-making processes and view them through the lens of computation, and on the other hand we study and address the limitations of artificial decision-making algorithms when deployed in various societal contexts. The focus is on developing solutions through a combination of foundational work such as coming up with the right definitions, modeling, algorithms, and empirical evaluation. The current focus is on bias and privacy, with additional topics including robustness, polarization, and democratic representation. The grade will be based on class participation and a project. The project grade will be determined by a midterm and endterm report/presentation. The course has four primary modules: (1) Data: human-generated data; data collection and aggregation; (2) Decision-Making Algorithms: human decision-making algorithms; traditional algorithmic decision-making models and methods; machine learning-based decision-making models and methods; (3) Bias: socio-technical contexts and underlying computational problems; definitions of fairness; interventions for ensuring fairness; human biases through the lens of computation; privacy; need for definitions of privacy; differential privacy; beyond differential privacy; (4) Other topics: robustness; polarization; elections and social choice. Solid mathematical and programming background is necessary to enroll in this course. CPSC 365 and S&DS 251 are recommended.

**CPSC 565b, Theory of Distributed Systems** James Aspnes
Models of asynchronous distributed computing systems. Fundamental concepts of concurrency and synchronization, communication, reliability, topological and geometric constraints, time and space complexity, and distributed algorithms.

**CPSC 566a or b, Blockchain and Cryptocurrency** Staff
This course is an introduction to blockchain systems, such as Bitcoin and Ethereum. We begin with a brief history of blockchains and an overview of how they are being used today before launching into foundational topics, including distributed consensus, smart contracts, cryptographic building blocks from signatures to authenticated datastructures, and the economics of blockchains. We then cover advanced topics including the scalability and interoperability of blockchain systems and applications

such as "decentralized finance" (DeFi). The lectures and assignments engage students in both theoretical and applied aspects of blockchain systems. The course assumes background in various fundamental areas of CS, including discrete math, probability, algorithms, data structures, cryptography, and networks.

**CPSC 567b, Introduction to Cryptography** Katerina Sotiraki
This course introduces modern symmetric and public-key cryptography as well as their broad applications, both from a theoretical and practical perspective. There is an initial emphasis on fundamental cryptographic primitives (e.g., block ciphers, pseudorandom functions, pseudorandom generators, one-way functions), their concrete efficiency and implementation, as well as their security definitions and proofs. Ways of combining such primitives that lead to more complex objects used to secure today's internet (e.g., via TLS), such as key exchange, randomized encryption, message authentication codes, and digital signatures are also studied. The last part of the course is devoted to modern and more advanced applications of cryptography (some of which are deployed at scale today), such as authenticated data structures, zero-knowledge proofs, oblivious RAM, private information retrieval, secret sharing, distributed consensus, and cryptocurrencies. (e.g, Bitcoin).

**CPSC 568a, Computational Complexity** Dylan McKay
Introduction to the theory of computational complexity. Basic complexity classes, including polynomial time, nondeterministic polynomial time, probabilistic polynomial time, polynomial space, logarithmic space, and nondeterministic logarithmic space. The roles of reductions, completeness, randomness, and interaction in the formal study of computation.

**CPSC 570b, Artificial Intelligence** Stephen Slade
Introduction to artificial intelligence research, focusing on reasoning and perception. Topics include knowledge representation, predicate calculus, temporal reasoning, vision, robotics, planning, and learning.

**CPSC 573a, Intelligent Robotics Laboratory** Brian Scassellati
Students work in small teams to construct novel research projects using one of a variety of robot architectures. Project topics may include human-robot interaction, adaptive intelligent behavior, active perception, humanoid robotics, and socially assistive robotics.

**CPSC 574a or b, Computational Intelligence for Games** James Glenn
A seminar on current topics in computational intelligence for games, including developing agents for playing games, procedural content generation, and player modeling. Students read, present, and discuss recent papers and competitions, and complete a term-long project that applies some of the techniques discussed during the term to a game of their choice.

**CPSC 575a / ENAS 575a / INP 575a, Computational Vision and Biological Perception** Steven Zucker
An overview of computational vision with a biological emphasis. Suitable as an introduction to biological perception for computer science and engineering students, as well as an introduction to computational vision for mathematics, psychology, and physiology students.

**CPSC 576b / AMTH 667b / ENAS 576b, Advanced Computational Vision** Steven Zucker
Advanced view of vision from a mathematical, computational, and neurophysiological perspective. Emphasis on differential geometry, machine learning, visual psychophysics, and advanced neurophysiology. Topics include perceptual organization, shading, color, and texture.

**CPSC 577b, Natural Language Processing** Arman Cohan
Linguistic, mathematical, and computational fundamentals of natural language processing (NLP). Topics include part of speech tagging, Hidden Markov models, syntax and parsing, lexical semantics, compositional semantics, machine translation, text classification, discourse, and dialogue processing. Additional topics such as sentiment analysis, text generation, and deep learning for NLP.

**CPSC 578a, Computer Graphics** Theodore Kim
Introduction to the basic concepts of two- and three-dimensional computer graphics. Topics include affine and projective transformations, clipping and windowing, visual perception, scene modeling and animation, algorithms for visible surface determination, reflection models, illumination algorithms, and color theory.

**CPSC 579b, Advanced Topics in Computer Graphics** Julie Dorsey
An in-depth study of advanced algorithms and systems for rendering, modeling, and animation in computer graphics. Topics vary and may include reflectance modeling, global illumination, subdivision surfaces, NURBS, physically based fluids systems, and character animation.

**CPSC 581b, Introduction to Machine Learning** Alex Wong
This course focuses on fundamental topics in machine learning. We begin with an overview of different components of machine learning and types of learning paradigms. We introduce a linear function, discuss how one can train a linear function on a given dataset, and utilize it to tackle classification and regression problems. We then consider kernel methods to enable us to solve nonlinear problems. Additionally, we introduce the concept of generalization error and overfitting. We discuss the role of regularization and extend linear regression to ridge regression. We also cover topics in optimization, beginning from gradient descent and extending it to stochastic gradient descent and its momentum variant. We also cover the concept of alternating optimization and topics within it. We introduce the curse of dimensionality and discuss topics on dimensionality reduction. Finally, we conclude the course with neural networks: how to build them using the topics discussed, how to optimize them, and how to apply them to solve a range of machine learning tasks. Prerequisites: Courses in data structures and object-oriented programming (e.g. CPSC 223a or equivalent courses), foundational mathematical tools such as discrete math and linear algebra (e.g. CPSC 202 or equivalent courses), calculus (e.g. MATH 112, MATH 115, MATH 120, or equivalent courses), linear algebra (e.g. MATH 225, or equivalent courses), and artificial intelligence (e.g. CPSC 370/570). A background in statistics is useful but not required. Experience in programming with Python and familiarity with Google Colab and numerical and image processing packages (i.e. NumPy, SciPy) is helpful.

**CPSC 582b, Current Topics in Applied Machine Learning** David van Dijk
We cover recent advances in machine learning that focus on real-world data. We discuss a wide range of methods and their applications to diverse domains, such as finance,

health care, genomics, protein folding, drug discovery, neuroscience, and natural language processing. The seminar is based on a series of lectures by the instructor and guest lecturers, as well as student presentations. The latter are expected to be on recent publications from leading journals and conferences in the field and are followed by discussions. A final project involves the application of a machine-learning method to real-world data. Graduate students are required to work on projects, which are optional for undergraduates. Prerequisites: mathematical tools for computer science (CPSC 202 or equivalent course), linear algebra (MATH 222/MATH 225 or equivalent course), calculus (MATH 120 or equivalent course), or permission of the instructor; and basic coding knowledge (e.g., Python).

**CPSC 583a, Deep Learning on Graph-Structured Data** Rex Ying
Graph structure emerges in many important domain applications, including but not limited to computer vision, natural sciences, social networks, languages, and knowledge graphs. This course offers an introduction to deep learning algorithms applied to such graph-structured data. The first part of the course is an introduction to representation learning for graphs and covers common techniques in the field, including distributed node embeddings, graph neural networks, deep graph generative models, and non-Euclidean embeddings. The first part also touches upon topics of real-world significance, including auto-ML and explainability for graph learning. The second part of the course covers important applications of graph machine learning. We learn ways to model data as graphs and apply graph learning techniques to problems in domains including online recommender systems, knowledge graphs, biological networks, physical simulations and graph mining. The course covers many deep techniques (graph neural networks, graph deep generative models) catered to graph structures. We cover basic deep learning tutorials in this course. Knowledge of graphs as a data structure, and understanding of basic graph algorithms are essential for applying machine learning to graph-structured data. Familiarity with Python and important libraries such as Numpy and Pandas are helpful. A foundation of deep neural networks is highly recommended. Experience in machine Learning and Graph Theory are welcomed as well.

**CPSC 586b, Probabilistic Machine Learning** Andre Wibisono
This course provides an overview of the probabilistic frameworks for machine learning applications. The course covers probabilistic generative models, learning and inference, algorithms for sampling, and a survey of generative diffusion models. This course studies the theoretical analysis of the problems and how to design algorithms to solve them. This course familiarizes students with techniques and results in literature and prepares them for research in machine learning. Prerequisites: Knowledge of machine learning, linear algebra, probability, and calculus.

**CPSC 611a, Topics in Computer Science and Global Affairs** Joan Feigenbaum and Ted Wittenstein
This course focuses on "socio-technical" problems in computing and international relations. These are problems that cannot be solved through technological progress alone but rather require legal, political, or cultural progress as well. Examples include but are not limited to cyber espionage, disinformation, ransomware attacks, and intellectual-property theft. This course is offered jointly by the SEAS Computer Science Department and the Jackson School of Global Affairs. It is addressed to graduate students who are interested in socio-technical issues but whose undergraduate course

work may not have addressed them; it is designed to bring these students rapidly to the point at which they can do research on socio-technical problems. Prerequisites: Basics of cryptography and computer security (as covered in Yale's CPSC 467), networks (as covered in Yale's CPSC 433), and databases (as covered in Yale's CPSC 437) helpful but not required.

**CPSC 612b, Topics in Algorithmic Game Theory** Yang Cai
The course focuses on algorithms and the complexity of equilibrium computation as well as its connection with learning theory and optimization. As many recent machine learning approaches have moved from an optimization perspective to an "equilibration" perspective, where a good model is framed as the equilibrium of a game. The intersection of game theory, learning theory, and optimization is becoming increasingly relevant. The goal of the course is to cover the fundamentals and bring students to the frontier of this active research area. Prerequisite: A course in algorithms (CPSC 365 or 366) and a course in probability theory (MATH/S&DS 241). A course in algorithmic game theory (CPSC 455/555) is helpful but not required.

**CPSC 640b / AMTH 640b / MATH 640b, Topics in Numerical Computation** Vladimir Rokhlin
This course discusses several areas of numerical computing that often cause difficulties to non-numericists, from the ever-present issue of condition numbers and ill-posedness to the algorithms of numerical linear algebra to the reliability of numerical software. The course also provides a brief introduction to "fast" algorithms and their interactions with modern hardware environments. The course is addressed to Computer Science graduate students who do not necessarily specialize in numerical computation; it assumes the understanding of calculus and linear algebra and familiarity with (or willingness to learn) either C or FORTRAN. Its purpose is to prepare students for using elementary numerical techniques when and if the need arises.

**CPSC 646a, Combinatorial Optimization and Approximation Algorithms** Staff
The course covers the design and analysis of approximation algorithms via combinatorial techniques. We start with classical polynomial time combinatorial optimization problems, including matchings, flows, cuts, and submodular functions. In the latter half, we discuss techniques for designing approximation algorithms for NP-hard problems, including the primal-dual method, randomized rounding, iterative relaxations, and local search. Prerequisites: some background in algorithms and discrete mathematics as well as familiarity with linear programming.

**CPSC 648a, Quantum Codes and Applications to Complexity** Staff
The course covers the theory of quantum error correction and its applications to quantum complexity theory. We start with basic quantum codes and then progress towards more advanced code constructions, in particular good LDPC codes. In the later half, we discuss various intriguing applications of quantum codes in quantum complexity, in particular how they are used in NLTS construction. This course should be accessible to students without any background in quantum computing and complexity theory. Students with no such background are provided with additional reading material to catch up. Please reach out to the instructor if you have any questions.

**CPSC 690a or b, Independent Project I** Staff
By arrangement with faculty.

**CPSC 691a or b, Independent Project II** Staff
By arrangement with faculty.

**CPSC 692a, Independent Project** Holly Rushmeier
Individual research for students in the M.S. program. Requires a faculty supervisor and the permission of the director of graduate studies.

**CPSC 752b / CB&B 752b / MB&B 752b and MB&B 753b and MB&B 754b / MB&B 753b and MB&B 754b / MB&B 754b / MCDB 752b, Biomedical Data Science: Mining and Modeling** Mark Gerstein and Matthew Simon
Biomedical data science encompasses the analysis of gene sequences, macromolecular structures, and functional genomics data on a large scale. It represents a major practical application for modern techniques in data mining and simulation. Specific topics to be covered include sequence alignment, large-scale processing, next-generation sequencing data, comparative genomics, phylogenetics, biological database design, geometric analysis of protein structure, molecular-dynamics simulation, biological networks, normalization of microarray data, mining of functional genomics data sets, and machine-learning approaches to data integration. Prerequisites: biochemistry and calculus, or permission of the instructor.

**CPSC 776b, Topics in Industrial AI Applications** Xiuye (Sue) Chen
Techniques developed in AI research are now used in many industrial applications, ranging from voice assistants to scientific modeling to generative AI. The goal of this seminar is for students to acquire familiarity with current topics relevant to industry, and to apply related approaches to problems in their respective areas of expertise. Each year the course covers several different topics in industrial AI research, broadly defined. These topics may include edge ML, speech recognition, natural language processing, computer vision, ambient intelligence, generative AI, and applications to life sciences and healthcare. In most meetings, one or more key papers are discussed, and one student is chosen in advance to present the main ideas in the paper and guide the discussion. We also have guest speakers from industry to present or lead discussions on current industrial research topics. Periodically, we devote meeting sessions to discuss formulation of new research directions that leverage students' ongoing research in other areas. Grades are based in equal parts on discussion leadership, discussion participation, and research-problem formulation.

**CPSC 990a, Ethical Conduct of Research for Master's Students** Inyoung Shin
This course forms a vital part of research ethics training, aiming to instill moral research codes in graduate students of computer science, math, and applied math. By devling into case studies and real-life examples related to research misconduct, students will grasp core ethical principles in research and academia. The course also offers an opportunity to explore the societal impacts of research in computer science, math, and applied math. This course is designed specifically for first-year graduate students in computer science/applied math/math. Successful completion of the course necessitates in-person attendance on eight occasions; virtual participation will not fulfill this requirement. In cases where illness, job interviews, or unforeseen circumstances prevent attendance, makeup sessions will be offered. This course is 0 credits for YC students. 0 Course cr

**CPSC 991a / MATH 991a, Ethical Conduct of Research** Inyoung Shin
This course forms a vital part of research ethics training, aiming to instill moral research codes in graduate students of computer science, math, and applied math. By delving into case studies and real-life examples related to research misconduct, students grasp core ethical principles in research and academia. The course also offers an opportunity to explore the societal impacts of research in computer science, math, and applied math. This course is designed specifically for first-year graduate students in computer science, applied math, and math. Successful completion of the course necessitates in-person attendance on eight occasions; virtual participation does not fulfill this requirement. In cases where illness, job interviews, or unforeseen circumstances prevent attendance, makeup sessions are offered. 0 Course cr

**CPSC 992a, Academic Writing** Janet Kayfetz
This course is an intensive analysis of the principles of excellent writing for Ph.D. students and scientists preparing a range of texts including research papers, conference posters, technical reports, research statements, grant proposals, correspondence, science and industry blogs, and other relevant documents. We look at the components of rhetorical positioning in the development of a clear, interesting, and rigorous science research paper. Some of the sub-genres we analyze and practice include the introduction, literature review, methodology, data commentary, results/discussion, conclusion, and abstract. In addition to the research paper, we practice other types of texts including research statements, requests for funding, bio-data statements, and blogs. We also discuss how writers can develop content and fluency as well as strategies for redrafting and editing. Students receive detailed feedback on their writing with a focus on clarity, precision, tone, and readability. 0 Course cr